A Quantitative Study of Two Attacks [1]

Chiara Bodei, Michele Curti, Pierpaolo Degano [2]

Dipartimento di Informatica, Università di Pisa
Viale F. Buonarroti, 2, I-56127 Pisa, Italy.

Corrado Priami [3]

Dipartimento di Informatica e Telecomunicazioni, Università di Trento
Via Sommarive, 14 – 38050 Povo (TN), Italy.

September 28, 2004

Abstract

We use a special operational semantics which helps us in predicting quantitative measures on systems describing cryptographic protocols: We also consider a possible attacker. The transitions of the system carry enhanced labels. We assign rates to transitions by only looking at these labels. We then map transition systems to Markov chains and evaluate performance of systems, using standard tools.

1 Introduction

Cryptographic protocols, used in distributed systems for authentication and key exchange, are designed to guarantee security. The mechanisms used are always the result of a judicious balance between their cost and benefits. Performance and implementation costs must be carefully evaluated: the greater the threat of attacks, the greater the investment in security. In the end, security is not a state of perfection, but a process of risk management.

In [4], we proposed a first step towards the development of a single, formal design methodology that supports designers in analysing the performance of protocols, in terms of time overhead and resource consumption.

[1] Supported in part by the Information Society Technologies programme of the European Commission, Future and Emerging Technologies, under the IST-2001-32072 project DEGAS; the Danish SNF-project LoST.

[2] Email: {chiara,curtim,degano}@di.unipi.it

[3] Email: priami@science.unitn.it

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

We would like to extend [4], in order to include attackers into the picture so to evaluate the costs of particular sequences of successful attacks. As a matter of fact, estimating the attacker efforts can provide useful information for establishing the acceptability of a particular protocol and of its security in terms of its resilience. The idea is providing users with a formal way to evaluate the trade-off between the need to manage the risk of attacks and the need to keep costs low.

Actually, the attacker is a principal that does not play fair. According to the traditional Dolev-Yao [12] model, the attacker is supposed to have control over the communication network. It is therefore able to monitor the traffic and intercept a message, to create a new message, using parts of old ones, to inject new messages and to substitute its messages for legitimate ones. These actions can be easily modeled in a process algebraic framework.

The starting point of our approach is the narration of a specific protocol and of a documented attack to it. We detail the narration in the process algebra LySa [3], that makes it possible to specify the behaviour of both the legitimate participants and of the attacker in terms of processes running in parallel. The quantitative analysis applied on this specification gives us an estimate of the effort made by the attacker in this particular case.

The Dolev-Yao model leaves out those operations, classified as computationally hard, that an attacker can try to exploit, such as guessing keys and breaking cryptography systems. The attacker can indeed exploit the weaknesses of the encryption algorithm to obtain the plaintext, by suitably manipulating the ciphertext bit by bit. To take into account also this kind of actions, we need to extend our standard semantics (see Cervasato [7] and Meadows [17]): messages are not atomic objects, rather they are sequences of packets in turn made of bits. Communication and encryption/decryption are not atomic operations, as well.

Following [4], we describe protocols and their attacks using the process algebra LySa [3] with an enhanced semantics (along the lines of [11]) and we associate a cost with each transition, as proposed in [19].

More precisely, the enhanced operational semantics provides transitions with enhanced labels, from which it is possible to mechanically derive rates. Once rates have been assigned, it is easy to derive the Continuous Time Markov Chain associated with the transition system. From its stationary distribution, if any, we evaluate the performance of the process in hand.

2 Case Study

A balance between security and cost is particularly crucial for wireless LAN (WLAN), where the risks due to portability of wireless devices increase w.r.t. the risks of usual wired networks. A common way of protecting communications in a WLAN is by encrypting and decrypting messages with Wired Equivalent Privacy (WEP), the cryptographic algorithm for the IEEE 802.11 standard.

WEP should offer some security guarantees in terms of confidentiality, data integrity and access to the wireless network.

The algorithm relies on a secret key K shared between a mobile station (MS), e.g. a laptop with a wireless card, and an Access Point (AP), i.e. a base station. This key, exchanged apart, is used to protect the body of a transmitted message M. On the net it will pass the corresponding ciphertext $\{M\}_K$, together with the Initialization Vector v, as described below. To encrypt the message M, the sender proceeds as follows:

- M is checksummed (using the CRC-32 algorithm) to obtain $c(M)$; M and $c(M)$ are concatenated to obtain the plaintext $P = \langle M, c(M)\rangle$;
- an Initialization Vector (24-bit field) v is picked; the RC4 algorithm (a stream cipher) generates a keystream $RC4(v, K)$; the ciphertext C results from applying the exclusive-or (XOR, denoted by $\oplus$) to the plaintext and the keystream, i.e. $C = P \oplus RC4(v, K)$.

The whole frame of data, transmitted on the net, is therefore $\langle v, C\rangle$. To decrypt C the receiver reverses the encryption process:

- given v, he computes $RC4(v, K)$ and XORs the received message C against $RC4(v, K)$ to recover P, i.e. $C \oplus RC4(v, K) = (P \oplus RC4(v, K)) \oplus RC4(v, K) = P = \langle M, c(M)\rangle$;
- then, the receiver checks whether the checksums are equal. If so, the message is accepted.

Unfortunately, the WEP algorithm has a lot of flaws, due to its inappropriate implementation and combination of RC4 and CRC-32 (see e.g. [5,2]). Consequently, it is vulnerable to several attacks, heavily based on the cryptoanalysis. The main reasons for this include:

keystream reuse: given two ciphertexts, it is easy to obtain the XOR of the two corresponding plaintexts, due to algebraic properties of $\oplus$. In addition, traffic analysis can lead to discover plaintexts. Using an Initialization Vector seems to help, since it produces a different RC4 for each packet. Instead, the small space of initialization vectors claims for a highly probable reuse.

key reuse and management: usually, a unique key is shared between all mobile stations and access points.

poor packet integrity: the integrity checksum field used by WEP is implemented as a CRC-32 checksum. CRC is designed to detect random errors in a message, but it cannot prevent attacks. Therefore message integrity cannot be completely guaranteed: an encrypted message can be modified, without disrupting its checksum.

A couple of attacks follow that exploit the above vulnerabilities. It happens that neither involves disclosing the key. The first is an attack to the WLAN authentication protocol, a challenge-response protocol used by a mobile station to associate with an access point to authenticate itself.

$$(Prot_1)\quad \begin{array}{l} 1.\ MS \rightarrow AP : req, MS \\ 2.\ AP \rightarrow MS : N \\ 3.\ MS \rightarrow AP : v, \{N\}_K \\ 4.\ AP \rightarrow MS : Ack \end{array}$$

When the MS requests access (through the message req, MS) to the wireless network, the AP sends a challenge N; MS receives and encrypts it and then sends it back. AP decrypts the received packet and checks the payload against the challenge: if the two match, the authentication ends up successfully.

The observation of a legitimate authentication sequence it is sufficient to mount an authentication spoofing attack, because it directly provides the Initialization Vector and implicitly reveals the corresponding keystream. In fact $\{N\}_K$ is actually in the form $C = P \oplus RC4(v, K)$, with $P = \langle N, C(N)\rangle$. The attacker that knows both C and P, exploits the fact that $P \oplus C = P \oplus P \oplus RC4(v, K) = RC4(v, K)$, thus acquiring the knowledge of $RC4(v, K)$ besides the one of v. This is enough for the attacker to produce $\{N'\}'_K$ as $\langle N', C(N')\rangle \oplus RC4(v, K)$, without knowing K, and therefore to be authenticated. The attack sequence is

$$(Att_1)\quad \begin{array}{ll} 2.\ AP \rightarrow E(MS) : & N \\ 3.\ MS \rightarrow E(AP) : & v, \{N\}_K \\ 2'.\ AP \rightarrow E : & N' \\ 3'.\ E \rightarrow AP : & v, \{N'\}'_K \end{array}$$

The second attack, called IP redirection attack, is used when the access point acts as an IP router for mobile stations. AP decrypts all the packets sent by a mobile station and sends them (in clear) on Internet. The attacker tricks the access point, i.e. the legitimate decryptor: the access point will kindly decrypt for the attacker. In detail, MS sends AP a message in the form $\{H, M\}_K$, where the header of the ciphertext represents the intended destination IP address H. AP decrypts the packet and sends M to H. The attacker first intercepts the packet encrypted for H and then modifies it so that it is sent to EH, i.e. an IP address controlled by the attacker. The access point decrypts the forged packet and sends its content to the new destination in clear. Again it is not necessary to break the key. Modifying the address part of the packet is quite easy. The attacker has only to guess the header H, whose position is at a fixed length from the start of the packet. Figuring out H is not difficult. Just knowing H, it is sufficient to perform a selective modification on the ciphertext. The attack sequence is

$$
\begin{array}{ll}
 & 1.\ MS \to E(AP) : \{H, M\}_K \\
(Att_2) & 1'.\ E(MS) \to AP : \{EH, M\}_K \\
 & 2'.\ AP \to EH : \quad M
\end{array}
$$

Below we introduce our technique, and then we proceed with the application of our framework to the couple of attacks above.

3 LySA and its Enhanced Semantics

We briefly survey the process calculus LySA and give an intuitive introduction to its semantics.

3.1 The calculus

The LySA calculus [3] is based on the π- [18] and Spi-calculus [1], but differs from these essentially in two aspects: (i) the absence of channels: there is only one global communication medium to which all processes have access; (ii) the tests associated with input and decryption are naturally expressed using pattern matching. Below, we assume the reader familiar with the basics of process calculi.

Syntax.

The syntax consists of terms $E \in \mathcal{E}$ and processes $P \in \mathcal{P}$, where $\mathcal{N}$ and $\mathcal{X}$ denote sets of names and variables, respectively.

$E ::=$ *terms*

a	name ($a \in \mathcal{N}$)
x	variable ($x \in \mathcal{X}$)
$\{E_1, \cdots, E_k\}_{E_0}$	symmetric encryption ($k \geq 0$)

$P ::=$ *processes*

0	nil
$out.P$	output
$in.P$	input (with matching)
$P_1 \mid P_2$	parallel composition
$(\nu\, a)P$	restriction
dec in P	symmetric decryption (with matching)
$A(y_1, \ldots, y_n)$	constant definition

where the we used the following abbreviations, that we will use hereafter:

$out \triangleq \langle E_1, \cdots, E_k \rangle$

$in \triangleq (E'_1, \cdots, E'_j; x_{j+1}, \cdots, x_k)$,

$dec \triangleq \mathsf{decrypt}\ E\ \mathsf{as}\ \{E_1, \cdots, E_j; x_{j+1}, \cdots, x_k\}_{E_0}$.

Intuitively, $\mathsf{0}$ represents the null inactive process. The operator $|$ describes parallel composition of processes. The operator (νa) acts as a static declaration (i.e. a binder) for the name a in the process P the restriction prefixes. Restriction is therefore used to create new names such as nonces or keys. The process $\langle E_1, \cdots, E_k \rangle.\, P$ sends $E_1, \cdots, E_k$ on the net and then continues like P. The process $(E_1, \cdots, E_j;\ x_{j+1}, \cdots, x_k).\, P$ attempts to receive the tuple $E'_1, \cdots, E'_k$ and to continue as $P[E_{j+1}/x_{j+1}, \ldots, E_k/x_k]$, provided that $E_i = E'_i$ for all $i \in [1, j]$. The intuition is that the matching succeeds when the first j values E'_i pairwise correspond to the values E_i, and the effect is to bind the remaining $k - j$ values to the variables $x_{j+1}, \cdots, x_k$. Note that, syntactically, a semi-colon separates the components where matching is performed from those where only binding takes place. The same simple form of patterns is also used for decryption (see [6] for a more flexible choice). In fact, the process $\mathsf{decrypt}\ E\ \mathsf{as}\ \{E_1, \cdots, E_j;\ x_{j+1}, \cdots, x_k\}_{E_0}\ \mathsf{in}\ P$ attempts to decrypt $E = \{E'_1, \cdots, E'_k\}_{E'_0}$ with the key E_0. Whenever $E_i = E'_i$ for all $i \in [0, j]$, the process behaves as $P[E_{j+1}/x_{j+1}, \ldots, E_k/x_k]$. Finally, an agent is a static definition of a parameterised process. Each agent identifier A has a unique defining equation of the form $A(\tilde{y}) = P$, where $\tilde{y}$ denotes a tuple $y_1, \ldots, y_n$ of distinct names (the only ones) occurring free in P.

1	$Sys_1 = (\nu K)(\nu v)(AP \mid MS)$	K shared key and v initialization vector
2	$AP = (AP, req;\ z_{MS}).\, AP'$	AP receives and checks msg (1)
3	$AP' = (\nu N)(\langle AP, z_{MS}, N \rangle.\, AP'')$	AP sends msg (2)
4	$AP'' = (z_{MS}, AP, v;\ z_{enc}).\, AP'''$	AP receives and checks msg (3)
5	$AP''' = \mathsf{decrypt}\ z_{enc}\ \mathsf{as}\ \{N;\ \}_K\ \mathsf{in}\ AP''''$	AP decrypts the enc in msg (3)
6	$AP'''' = \langle AP, z_{MS}, ack \rangle.\, AP$	AP sends msg (4)
7	$MS = \langle AP, req, MS \rangle.\, MS'$	MS sends msg (1)
8	$MS' = (AP, MS;\ x_N).\, MS''$	MS receives and checks msg (2)
9	$MS'' = \langle MS, AP, v, \{x_N\}_K \rangle.\, MS'''$	MS sends msg (3)
10	$MS''' = (AP, MS, ack;\ ).\, MS$	MS receives and checks msg (4)

Table 1
WEP Authentication Protocol

Example

The LySA specification of the WEP authentication protocol is in Table 1, where each message appears endowed with the name of the sender and the receiver, for the sake of clarity. The right part has a concise explanation of the action on the left, in terms of the number of the message (called msg, while enc stands for an encrypted term) in the protocol narration $Prot_1$, presented in Section 2.
The whole system is given by the parallel composition ($|$) of the two processes AP and MS. Each of them performs a certain number of actions and then re-starts again. They share the key K and the initialization vector v.

We describe a part of a computation that arises from Sys_1 to show the basic forms of LySA dynamics, i.e. communication and decryption.

$1\ (\nu K)(\nu v)(AP, req;\ z_{MS}).\ AP' | \langle AP, req, MS\rangle.\ MS' \longrightarrow (\nu K)(\nu v)(AP'[MS/z_{MS}] | MS')$

$2\ (\nu K)(\nu v)((\nu N)(\langle AP, MS, N\rangle.\ AP'')[MS/z_{MS}] | MS')$

$\longrightarrow (\nu K)(\nu v)(\nu N)(AP''[MS/z_{MS}] | MS''[N/x_N])$

$3\ (\nu K)(\nu v)(\nu N)((z_{MS}, AP, v;\ z_{enc}).\ AP'''[MS/z_{MS}] | \langle MS, AP, v, \{N\}_K\rangle.\ MS''')$

$\longrightarrow (\nu K)(\nu v)(\nu N)(AP'''[MS/z_{MS}, \{N\}_K/z_{enc}] | MS''')$

$3'\ (\nu K)(\nu v)(\nu N)(\mathsf{decrypt}\ \{N\}_K\ \mathsf{as}\ \{N;\ \}_K\ \mathsf{in}\ AP''''[MS/z_{MS}] | MS''')$

$\longrightarrow (\nu K)(\nu v)(\nu N)(AP''''[MS/z_{MS}] | MS''')$

$4\ (\nu K)(\nu v)(\nu N)(\langle AP, MS, ack\rangle.\ AP | (AP, MS, ack;\ ).\ MS \longrightarrow Sys_1$

In the first transition (1), the principal AP receives (see line 2 in Tab. 1) the message $\langle AP, req, MS\rangle$ sent by MS. The first two terms are AP, req as required by the input prefix in the form $(AP, req;\ MS)$.. Consequently, the variable z_{MS} is bound to MS, within the continuation AP', i.e. to the last term of the received message MS. Similarly for the second, third and fifth transitions $(2, 3, 4)$. In the fourth transition $(3')$, the process AP''' that after the substitution is $\mathsf{decrypt}\ \{N\}_K\ \mathsf{as}\ \{N;\ \}_K\ \mathsf{in}\ AP''''$ attempts to decrypt $\{N\}_K$ with the key K (line 5 in Tab. 1) and checks whether the decrypted term is the nonce N.

3.2 Enhanced Semantics

Once the protocol has been specified in LySA, our semantics associates a label with each transition, in particular to each communication and to each decryption. To this aim, we use a very concrete version of semantics, called *enhanced*, in the style of [10,11]. Our enhanced semantics for LySA is a reduction semantics, built on top of the standard reduction semantics [3], where both processes and transitions are annotated with labels. Labels encode information about the actions and about the syntactic context in which actions take place. The encoding of the context for each prefix of a given process, is obtained by an-

notating each of them with a string of tags ϑ, called *context label*. This string essentially records which is the syntactic position of the prefix with respect to the parallel composition nesting: a tag $\|_0$ ($\|_1$, respectively) is used for the left (for the right, respectively) branch of a parallel composition. In order to do this, we exploit the abstract syntax tree of processes, built using the binary parallel composition as operator. The labels θ that enrich transitions are called *enhanced labels*. The enhanced label for a communication is in the form $\theta = \langle \vartheta_O out, \vartheta_I in \rangle$ and records the input and output perfixes that lead to the transition. Similarly, a transition for a decryption has a label in the form $\langle \vartheta dec \rangle$.

Example (cont'd)

Back to our example, consider the label of the first transition of Sys_1: $\theta_0 = \langle \|_0(AP, req; z_{MS}), \|_1\langle AP, req, MS\rangle\rangle$. The context label of the prefixes of AP is simply $\|_0$, while the one of the prefixes of MS is $\|_1$. This corresponds to the fact that AP and MS occur in the left (right, respectively) branch of the parallel composition $(AP|MS)$. For lack of space, we omit drawing the corresponding transition system.

The enhanced labels of all the transitions of Sys_1 are below. While, in general, the same label can annotate different transitions, in our example, there is only one label for each transition. Therefore, we feel free to use hereafter the label θ_i for the i-th transition.

- $\theta_0 = \langle \|_0(AP, req; z_{MS}), \|_1\langle AP, req, MS\rangle\rangle$;
- $\theta_1 = \langle \|_0(AP, MS; x_N), \|_1\langle AP, MS, N\rangle\rangle$;
- $\theta_2 = \langle \|_0(MS, AP, v; z_{enc}), \|_1\langle MS, AP, v, \{N\}_K\rangle\rangle$;
- $\theta_3 = \langle \|_0\ decrypt\ \{N\}_K\ as\ \{N;\}_K\rangle$
- $\theta_4 = \langle \|_0(AP, MS, ack;), \|_1\langle AP, MS, ack\rangle\rangle$

4 Stochastic Analysis

We are now ready to give the intuition on how to derive the costs of transitions by inspecting enhanced labels, following [19]. This information is sufficient to extract the necessary quantitative information to obtain the Continuous Time Markov Chains (CTMC). In general, by "cost" we mean any measure that affects quantitative properties of transitions: here, we intend the time the system is likely to remain within a given transition. Therefore, we specify the cost of a protocol in terms of the time overhead due to its primitives (along the same lines as [20]). The cost of (the component of) the transition depends on both the current action and on its context. This is vindicated in [19] as follows. Since the semantics of a language specifies its abstract machine, the context in which an action occurs represents the run-time support routines that the target machine performs in order to fire that action.

First, we intuitively present the main factors that influence the costs of actions and those due to their context.

- The cost of a *communication* depends on the costs of the input and output components. In particular, the cost of an
  - *output* depends on the size of the message and on the cost of each term of the message sent, in particular on its encryptions.
  - *input* depends on the size of the message and on the cost of checks needed to accept the message.

  Actually, the two partners independently perform some low-level operations locally to their environment, each of which leads to a delay. Since communication is synchronous and handshaking, the overall cost corresponds to the cost paid by the slower partner.
- The cost of both *encryption* and *decryption* depends on the sizes of the cleartext and ciphertext, respectively; the complexity of the algorithm that implements it; the cipher mode adopted; the kind of the key (short/long, short-term/long-term). The length of the key is important: usually, the longer the key, the greater the computing time. In addition , the cost for *decryption* depends on the cost of the checks needed to accept the decryption.
- The cost of *parallel composition* is evaluated according to the number of available processors and to the speed of system clock.

For simplicity, here we simply ignore the costs for other primitives, e.g. restriction or constant invocation; see [19] for a complete treament.

To define a cost function, we start by considering the execution of each action on a dedicated architecture that only has to perform that action, and we estimate the corresponding duration with a fixed rate r. Then we model the performance degradation due to the run-time support. In order to do that, we introduce a scaling factor for r in correspondence with each routine called by the implementation of the transition θ under consideration. Actually, we just propose a format for these functions, with parameters to be instantiated on need. Note that these parameters depend on the target machine. For instance, in a system where the cryptographic operations are performed at very high speed (e.g. on a cryptographic accelerator), but with a slow link (low bandwidth), the time will be low for encryptions and high for communication; vice versa, in a system offering a high bandwidth, but poor cryptography resources.

Technically, we interpret costs as parameters of exponential distributions (general distributions are also possible (see [21]), as they depend on enhanced labels, only). The *rate* associated with the transition is the parameter which identifies the exponential distribution of the duration times of the transition, as usual in in stochastic process algebras (e.g. [14,13]). An exponential distribution with rate r is a function $F(t) = 1 - e^{-rt}$, where t is the time parameter. The shape of $F(t)$ is a curve which grows from 0 asymptotically approaching

1 for positive values of its argument t. The parameter r determines the slope of the curve. The greater r, the faster $F(t)$ approaches its asymptotic value. The probability of performing an action with parameter r within time x is $F(x) = 1 - e^{-rx}$, so r determines the time, Δt, needed to have a probability near to 1.

4.1 Cost Functions

We define in a few steps the function that associates rates with communication and decryption transitions, or, more precisely, with their enhanced labels. We first give the auxiliary function $f_E : \mathcal{E} \to \mathbb{R}^+$ that estimates the effort needed to manipulate terms $E \in \mathcal{E}$.

$$f_E(a) = size(a)$$
$$f_E(\{E_1, \ldots, E_k\}_{E_0}) = f_{enc}(f_E(E_1), ..., f_E(E_1), kind(E_0))$$

The size of a name a ($size(a)$) matters. For an encrypted term, we use the function f_{enc}, which in turn depends on the estimate of the terms to encrypt and on the kind of the key (represented by $kind(E_0)$), i.e. on its length and on the corresponding cryptosystem.

Then we assign costs to actions in $\{in, out, dec\}$. Formally, the function $\$_\alpha : \{in, out, dec\} \to \mathbb{R}^+$ is defined as

$$\$_\alpha(\langle E_1, \ldots, E_k \rangle) = f_{out}(f_E(E_1), ..., f_E(E_1), bw)$$
$$\$_\alpha((E_1, \ldots, E_j; x_{j+1}, \ldots, x_k)) = f_{in}(f_E(E_1), ..., f_E(E_j), match(j), bw)$$
$$\$_\alpha(\mathsf{decrypt}\ E\ \mathsf{as}\ \{E_1, \cdots, E_j; x_{j+1}, \cdots, x_k\}_{E_0}) =$$
$$f_{dec}(f_E(E), kind(E_0), match(j))$$

The functions f_{out} and f_{in} define the costs of the routines which implement the send and receive primitives. Besides the implementation cost due to their own algorithms, the functions above depend on the bandwidth of the communication channel (represented by bw) and the cost of the exchanged terms, in turn computed by the auxiliary function f_E. Furthermore, the cost of an input depends on the number of tests or matchings required (represented by $match(j)$). Finally, the function f_{dec} represents the cost of a decryption. It depends on the manipulated terms ($f_E(E)$), on the kind of key ($kind(E_0)$) and on the number of matchings ($match(j)$).

We now consider the context in which the actions occur. To determine the slowing factor due to parallel composition, we associate a cost to each context label ϑ, as expressed by the function $\$_l : \{\|_0, \|_1\}^* \to (0, 1]$. Parallel composition is evaluated according to the number np of processors available, and on the number of processes that run on them, represented by the number of tags ($|\cdot|$) in the context label. Another factor is given by the speed of $clock$, the system clock.

$$\$_l(\vartheta) = f_{||}(np, |\vartheta|, clock),\ i = 0, 1$$

Finally, we give the function $\$: \Theta \to \mathbb{R}^+$ that associates rates with enhanced labels.

$$\$(\langle \vartheta_O out, \vartheta_I in \rangle) = \min\{\$_l(\vartheta_O) \cdot \$_\alpha(out), \$_l(\vartheta_I) \cdot \$_\alpha(in)\}$$
$$\$\langle \vartheta dec \rangle = \$_l(\vartheta) \cdot \$_\alpha(dec)$$

As mentioned above, the two partners independently perform some low-level operations locally to their environment, represented by the two context labels ϑ_O and ϑ_I. Each of these labels leads to a delay ($\$_l(\vartheta_O)$ and $\$_l(\vartheta_I)$, respectively) in the rate of the corresponding action ($\$_\alpha(out)$ and $\$_\alpha(in)$, respectively). Therefore, the cost paid by the slower partner corresponds to minimum of those of the operations independently performed by the participants, in isolation. Indeed the lower the cost, i.e. the rate, the greater the time needed to complete an action and hence the slower the speed of the transition occurring [4].

Note that we do not fix the actual cost function: we only propose for it a possible set of parameters that reflect some features of a somewhat idealized architecture and of a particular cryptosystem. Although very abstract, this suffices to make our point. A precise instantiation comes with the refinement steps from specification to implementations as soon as actual parameters become available.

Example (cont'd)

We now associate a rate to each transition in the transition system Sys_1. For the sake of simplicity, we assume that each principal has enough processing power and then we can map each ϑ to 1. We could vary this value considering e.g. differences in the speed of clock for the two processes. The instantiation of the cost functions given make use of the following parameters to represent the time spent while performing the corresponding action on a single term.

- `e` and `d` for encrypting and for decrypting,
- `s` and `r` for sending and for receiving,
- `m` for pattern matching.

The functions are:

$$f_E(a) = 1$$
$$f_E(\{E_1, \ldots, E_k\}_{E_0}) = \frac{\mathtt{e}}{\mathtt{s}} \cdot \sum_{i=1}^{k} f_E(E_i) + \sum_{i=1}^{k} f_E(E_i)$$
$$\$_\alpha(\langle E_1, \ldots, E_k \rangle) = \frac{1}{\mathtt{s} \cdot \sum_{i=1}^{i} f_E(E_i)}$$
$$\$_\alpha((E_1, \ldots, E_j; x_{j+1}, \ldots, x_k)) = \frac{1}{\mathtt{r} \cdot k + \mathtt{m} \cdot j}$$
$$\$_\alpha(\mathsf{decrypt}\ E\ \mathsf{as}\ \{E_1, \cdots, E_j; x_{j+1}, \cdots, x_k\}_{E_0}) = \frac{1}{\mathtt{d} \cdot k + \mathtt{m} \cdot j}$$

[4] The smaller r, the slower $F(t) = 1 - e^{-rt}$ approaches its asymptotic value.

Since transmission is usually more time-consuming than the corresponding reception, the rate of a communication, will always be that of output. The rate of the first transition in Sys_1 is $c_0 = \$(\theta_0) = \frac{1}{3\mathtt{s}}$ and corresponds to $min\{(\$_l(\|_0) \cdot \$_\alpha(\langle\langle AP, req; z_{MS}\rangle\rangle), \$_l(\|_1) \cdot \$_\alpha(\langle AP, req, MS\rangle)\} = min\{\frac{1}{3\mathtt{s}}, \frac{1}{3\mathtt{r}+\mathtt{m}}\}$. The other rates are $c_i = \$(\theta_i)$, of the transitions with labels θ_i in Sys_1 are:

$$c_0 = c_1 = c_4 = \frac{1}{3\mathtt{s}}, \quad c_2 = \frac{1}{4\mathtt{s}+\mathtt{e}}, \quad c_3 = \frac{1}{\mathtt{d}+\mathtt{m}}$$

4.2 Markov Chains and Performance Measures

Our first step is obtaining a Continuous Time Markov Chain (CTMC) from a transition system. Then, we shall calculate the actual performance measure, e.g. the throughput or utilization of a certain resource.

We use the rates of transitions computed Subsection 4.1, to transform a transition system T into its corresponding $CTMC(T)$: a state is associated with each node of the transition system, while the transitions between states are defined by the arcs.

Actually, the rate $q(T_i, T_j)$ at which a system changes from behaving like process T_i to behaving like T_j is the sum of the single rates of all the possible transitions from T_i to T_j. Note that $q(T_i, T_j)$ coincides with the off-diagonal element q_{ij} of the generator matrix of the CTMC, namely $\mathbf{Q}$. Recall that a CTMC can be seen as a directed graph and that its matrix $\mathbf{Q}$ (apart from its diagonal) represents its adjacency matrix. Hence, hereafter we will use indistinguishably CTMC and its corresponding $\mathbf{Q}$ to denote a Markov chain. More formally, the entries of the generator matrix $\mathbf{Q}$ are defined as

$$(1) \quad q_{ij} = \begin{cases} q(T_i, T_j) = \sum\limits_{T_i \xrightarrow{\theta_k} T_j} \$(\theta_k) & \text{if } i \neq j \\ -\sum\limits_{\substack{j=0 \\ j \neq i}}^{n} q_{ij} & \text{if } i = j \end{cases}$$

Example (cont'd)

Consider again the transition system Sys_1 which is finite and has a cyclic initial state. These features ensure the existence of its stationary distribution. The *stationary probability distribution* of a CTMC is $\Pi = (X_0, \ldots, X_{n-1})$ such that Π solves the matrix equation $\Pi^T \mathbf{Q} = \mathbf{0}$ and $\sum_{i=0}^{n} X_i = 1$. We derive the following generator matrix $\mathbf{Q}$ of $CTMC(Sys_1)$.

$$\mathbf{Q} = \begin{bmatrix} -c_0 & c_0 & 0 & 0 & 0 \\ 0 & -c_1 & c_1 & 0 & 0 \\ 0 & 0 & -c_2 & c_2 & 0 \\ 0 & 0 & 0 & -c_3 & c_3 \\ c_4 & 0 & 0 & 0 & -c_4 \end{bmatrix}$$

The stationary distribution of the Markov chain Π for Sys_1 is

$$\Pi = \left[\frac{3\mathtt{s}}{A}, \frac{3\mathtt{s}}{A}, \frac{4\mathtt{s}+\mathtt{e}}{A}, \frac{\mathtt{d}+3\mathtt{m}}{A}, \frac{3\mathtt{s}}{A}\right] \quad \text{where } A = 13\mathtt{s}+\mathtt{e}+\mathtt{d}+3\mathtt{m}$$

Evaluating the Performance

In order to define performance measures for a process T, we define a *reward structure* associated with T, following [15,14,8]. Usually, a reward structure is simply a function that associates a reward with any state passed through in a computation of T. For instance, when calculating the utilisation of a resource, we assign value 1 to any state in which the use of the resource is enabled (typically the source of a transition that uses the resource). All the other states earn the value 0. Instead we use a slightly different notion, where rewards are computed from rates of transitions [19]. To measure instead the throughput of a system, i.e. the amount of useful work accomplished per unit time, a reasonable choice is to use as nonzero reward a value equal to the rate of the corresponding transition.

Example (cont'd)

The throughput for a given activity is found by first associating a transition reward equal to the activity rate with each transition. In our systems each transition is fired only once. Also, the graph of the corresponding CTMC is cyclic and all the labels represent different activities. This amounts to saying that the throughput of all the activities is the same, and we can freely choose one of them to compute the throughput of Sys_1. Thus we associate a transition reward equal to its rate with the last communication and a null transition reward with all the others communications. From them, we compute the reward structures as $\rho = (0, 0, 0, 0, c_4)$ for Sys_1. The total reward R of the system amounts then to $\frac{3\mathtt{s}}{13\mathtt{s}+\mathtt{e}+\mathtt{d}+3\mathtt{m}} \cdot \frac{1}{3\mathtt{s}} = \frac{1}{13\mathtt{s}+\mathtt{e}+\mathtt{d}+3\mathtt{m}}$. To use this performance measure, it is necessary to instantiate our parameters under various hypotheses, depending on several factors, such as the network load, the packet size, the number of mobile stations and so on. What it is known is that the the cost for encryption and decryption could be considered equivalent (i.e. $\mathtt{e} = \mathtt{d}$) and that it is much bigger than the one for matching $\mathtt{m}$ and the one for sending $\mathtt{s}$. Roughly, this means that the total reward can be approximated by $\frac{1}{2\mathtt{d}}$, thus confirming the idea that cryptographic mechanisms could weak the network performance.

5 Costs of Attacks

We are interested here in modelling the costs of both the attacks introduced in Section 2 and therefore we need to consider in our syntax also the extra-actions, in which an attacker can exploit its computational power and in its capability of guessing.

The First Attack

To model the attack to the WEP authentication protocol, we use a special construct called here *emulated encryption*. As explained in Section 2, if the attacker knows a nonce and its encryption, even though it ignores the key, it can obtain the encryption of any other nonce it knows. More precisely it is able to produce the same bit string that can be obtained by encrypting with the proper key, just by XOR-ing the collected information. Actually, we treat an abstraction of this action, i.e. we neither introduce an explicit construct for the XOR operation and nor we explicitly handle $RC4(v, K)$. To illustrate this level of abstraction, we mimick the usual induction rules used to show the capabilities of the Dolev-Yao attacker, in terms of its knowledge $\mathcal{K}$.

$$\frac{\mathcal{K} \models N, \{N\}_K, N'}{\mathcal{K} \models \{\{N'\}\}_K}$$

So terms become:

$\tilde{E}$ ::= *extended terms*

E	standard term
$\{\{E\}\}_K$	emulated encryption

In Tab. 2, we present the specification Sys'_1 of the attack to the WEP authentication protocol Att_1, where the specification of AP and MS are as in Tab. 1, and the sequence of attack is represented by the process E. As described in Section 2, the attack consists in two phases: (i) the attacker E observes a whole authentication session; (ii) the attacker exploits the information acquired in (i) to authenticate itself, as the continuation $\overline{E}$. For simplicity, we chose to represent the observation made by E of each exchange of messages between the legitimate participants AS and MS in (i) as composed of two actions: (a) first E reads the message, by intercepting it from the net, and then (a') E sends it to the receiver, by injecting the message unaltered on the net. Note in the second phase the use of emulated encryption in the third message (line c'' in Tab. 2).

Because of space limitation, we do not show the 22-states transition system corresponding to Sys'_1. The context labels of processes are different, because the structure of processes now includes also E: $\|_0\|_0$ for AP, $\|_0\|_1$ for MS and $\|_1$ for E. The enhanced labels are similar to the ones seen for Sys_1. and the only interesting one is $\theta_2 = \langle \|_0\|_0(E, AP, v; z_{enc}), \|_1\langle E, AP, v, \{\{N'\}\}_K\rangle\rangle$.
It corresponds to the action specified in line c'' in Tab. 2, where E uses the emulated encryption to obtain the message $\langle E, AP, v, \{\{N'\}\}_K\rangle$. Its cost is $c_2 = \$(\theta_2) = \frac{1}{4\mathtt{s}+\mathtt{x}}$, where the parameter $\mathtt{x}$ corresponds to the time spent in producing the emulated encryption. Again, for the sake of simplicity, we assume that each principal has enough processing power and then we can map each ϑ to 1.

1	$Sys'_1 = (\nu K)(\nu v)(AP \mid MS) \mid E$	
a	$E = (AP, req, MS;\ ).\, E'$	E intercepts msg (1)
a'	$E^I = \langle AP, req, MS \rangle.\, E^{II}$	E injects msg (1)
b	$E^{II} = (AP, MS;\ y_N).\, E^{III}$	E intercepts msg (2)
b'	$E^{III} = \langle AP, MS, y_N \rangle.\, E^{IV}$	E injects msg (2)
c	$E^{IV} = (MS, AP;\ y_v, y_{enc}).\, E^V$	E intercepts msg (3)
c'	$E^V = \langle MS, AP, y_v, y_{enc} \rangle.\, E^{VI}$	E injects msg (3)
d	$E^{VI} = (AP, MS, ack;\ ).\, E^{VII}$	E intercepts msg (4)
d'	$E^{VII} = \langle AP, MS, ack \rangle.\, \overline{E}$	E injects msg (4)
a''	$\overline{E}^I = \langle AP, req, E \rangle.\, \overline{E}^{II}$	E sends msg (1)
b''	$\overline{E}^{II} = (AP, E;\ y_{N'}).\, \overline{E}^{III}$	E receives and checks msg (2)
c''	$\overline{E}^{III} = \langle E, AP, y_v, \{\{y_{N'}\}\}_K \rangle.\, \overline{E}^{IV}$	E sends msg (3)
d''	$\overline{E}^{IV} = (AP, E, ack;\ ).\, E$	E receives and checks msg (4)

Table 2
Attack to the WEP Authentication Protocol

We dispense with the details on the computation of the CTMC corresponding to the transition system, the technique is the same presented in Section 3. The generator matrix is $Q_1 =$

$$\begin{bmatrix}
-2c_0 & c_0 & 0 & 0 & 0 & c_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -c_1 & c_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -c_2 & c_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -c_3 & c_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
c_4 & 0 & 0 & 0 & -c_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -c_0 & c_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -c_1 & c_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -c_1 & c1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -c_2 & c_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -c_2 & c_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -c_3 & c_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -c_4 & c_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -c_4 & c_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2c_0 & c_0 & 0 & 0 & 0 & c_0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -c_1 & c_1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -c_{2p} & c_{2p} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -c_3 & c_3 & 0 & 0 & 0 & 0 \\
c_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -c_4 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -c_1 & c_1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -c_2 & c_2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -c_3 & c_3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & c_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -c_4
\end{bmatrix}$$

where $c_0 = c_1 = c_4 = \dfrac{1}{3\mathtt{s}}$, $\quad c_2 = \dfrac{1}{4\mathtt{s} + \mathtt{e}}$, $\quad c_3 = \dfrac{1}{\mathtt{d} + \mathtt{m}}$, $\quad c_{2p} = \dfrac{1}{4\mathtt{s} + \mathtt{x}}$

The corresponding stationary distribution is

$$\Pi_1 = \Big[\frac{3\mathtt{s}}{B}, \frac{3\mathtt{s}}{B}, \frac{4\mathtt{s}+\mathtt{e}}{B}, \frac{\mathtt{d}+\mathtt{m}}{B}, \frac{3\mathtt{s}}{B}, \frac{3\mathtt{s}}{B}, \frac{3\mathtt{s}}{B}, \frac{3\mathtt{s}}{B}, \frac{4\mathtt{s}+\mathtt{e}}{B}, \frac{4\mathtt{s}+\mathtt{e}}{B}, \frac{\mathtt{d}+\mathtt{m}}{B},$$

$$\frac{3\mathtt{s}}{B}, \frac{3\mathtt{s}}{B}, \frac{3\mathtt{s}}{B}, \frac{3\mathtt{s}}{B}, \frac{4\mathtt{s}+\mathtt{x}}{B}, \frac{\mathtt{d}+\mathtt{m}}{B}, \frac{3\mathtt{s}}{B}, \frac{3\mathtt{s}}{B}, \frac{4\mathtt{s}+\mathtt{e}}{B}, \frac{\mathtt{d}+\mathtt{m}}{B}, \frac{3\mathtt{s}}{B}\Big]$$

where $B = 59\mathtt{s} + 4\mathtt{e} + x + 4\mathtt{d} + 4\mathtt{m}$

Actually, our transition system presents two cyclic paths, where only one corresponds to the attack sequence. Thus we associate a transition reward equal to its rate with the last communication of this path and a null transition reward with all the others communications. The total reward R_1 of the system amounts then to $\frac{3\mathtt{s}}{B} \cdot \frac{1}{3\mathtt{s}} = \frac{1}{59\mathtt{s}+4\mathtt{e}+x+4\mathtt{d}+4\mathtt{m}}$.

Once instantiated the parameters, we can make more accurate performance considerations. Nevertheless, we can immediately observe that the throughput R of Sys_1 (the protocol specification, without the attacker) is approximatively four times greater than the throughput R_1 of Sys_1' (the protocol specification in the presence of the attack). Consequently, the attack and the authentication obtained by fraud does not appear so expensive. This result agrees with the ones achieved in literature [5,2] and quantitatively confirms that this form of authentication is completely unsatisfactory. As a matter of fact, the new versions of the WLAN offer more sophisticated (and more expensive too) forms of authentication.

The Second Attack

Similarly, we can evaluate the performance of the IP redirection attack. Also here, terms are extended with a new construct, called partially emulated encryption:

$\hat{E}$ ::= *extended terms*

E	standard term
$\{[E, E']\}_K$	partially emulated encryption

Again, we abstractly treat this encryption, as witnessed by the new induction rule for the attacker knowledge $\mathcal{K}$:

$$\frac{\mathcal{K} \models \{E, E'\}_K, E, E''}{\mathcal{K} \models \{[E'', E]\}_K}$$

The attacker is able to obtain the encryption $\{EH, M\}_K$ from $\{H, M\}_K$, without knowing the key K, but only knowing H and replacing it with EH.

Processes are instead extended with a new construct for guessing of a selected part of an encryption, assumed to lay always at the same position.

$\hat{P}$ ::= *extended processes*

P	standard process
$guess\ E\ as\ \{;x,?\}_K\ in\ P$	selective guessing

1	$Sys_3 = ((\nu K)(AP\|MS)\|H)\|(E\|EH)$	
1	$MS = \langle MS, AP, \{H, M\}_K\rangle.\, MS$	MS sends msg (1)
2	$H = (AP, H;\, z_M).\, H$	H receives and checks msg (2)
3	$AP = (MS, AP;\, x_{enc}).\, AP'$	AP receives and checks msg(1)
4	$AP' = \mathsf{decrypt}\ x_{enc}\ \mathsf{as}\ \{;\, x_d, x_M\}_K\ \mathsf{in}\ AP''$	AP decrypts its the enc in msg (1)
5	$AP'' = \langle AP, x_d, x_M\rangle.\, AP$	AP sends (2)
a	$E = ((MS, AP;\, y_{enc})).\, E'$	E intercepts msg (1)
a'	$E^I = guess\ y_{enc}\ as\{H;,?\}_K\ in\ E^{II}$	E guesses on msg (1)
a''	$E^{II} = \langle MS, AP, \{[EH, M]\}_K\rangle.\, E$	E injects msg (1')
b'	$EH = (AP, EH;\, z_M).\, EH$	EH receives and checks msg (2')

Table 3
Attack to the IP Redirection Protocol

In Tab. 3, we present the specification of the IP Redirection Protocol and of the attack Att_2, presented in Section 2, where the processes H and EH represent the processes that manage the corresponding IP addresses. To perform the attack, first E intercepts the encrypted packet sent by MS (line a in Tab. 3), then tries to figure out which is the original destination address H (line a'). Finally (line a'') E injects the modified encrypted packet, where the new destination address EH is one the attacker controls.

Also here we omit the transition system corresponding to Sys_2. The context labels of the main processes are: $\|_0\|_0\|_0$ for AP, $\|_0\|_0\|_1$ for MS, $\|_0\|_1$ for H, $\|_1\|_0$ for E, $\|_1\|_1$ for EH. The only interesting cases for enhanced labels correspond to

- the action specified in line a' in Tab. 3, where E tries to guess, whose cost is given by

$$c_3 = \$(\langle \|_1\|_0\ guess\ \{H, M\}_K\ as\{;H,?\}_K\rangle = \frac{1}{\mathsf{g}},$$

  where the parameter g corresponds to the time spent in this try.
- the action specified in line a'', where E uses the partially emulated encryption to obtain the message $\langle MS, AP, \{[EH, M]\}_K\rangle$, whose corresponding

cost is given by

$$c_4 = \$(\langle \|_0\|_0\|_0(MS, AP; \{[EH, M]\}_K), \|_1\|_0\langle MS, AP, \{[EH, M]\}_K\rangle\rangle) = \frac{1}{3\mathtt{s} + 2\mathtt{p}},$$

where the parameter $\mathtt{p}$ corresponds to the time spent in producing the partially emulated encryption.

The following generator matrix Q_2, has d_{ij} as generic element and is such that its diagonal elements d_{ii} are in the form $\sum_{\substack{j=0 \\ j\neq i}}^{n} d_{ij}$ and where

$$c_0 = \frac{1}{3\mathtt{s} + 2\mathtt{e}}, \quad c_1 = \frac{1}{2\mathtt{d}}, \quad c_2 = \frac{1}{3\mathtt{s}}, \quad c_3 = \frac{1}{\mathtt{g}}, \quad c_4 = \frac{1}{3\mathtt{s} + 2\mathtt{p}}$$

$$\begin{bmatrix}
-d_0 & c_0 & 0 & c_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -d_1 & c_1 & 0 & 0 & 0 & 0 & c_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
c_2 & 0 & -d_2 & 0 & 0 & 0 & 0 & 0 & c_0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -d_3 & c_3 & 0 & 0 & c_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -d_4 & c_4 & 0 & 0 & 0 & 0 & 0 & c_0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -d_5 & c_1 & 0 & 0 & c_0 & 0 & 0 & 0 & 0 & 0 \\
c_2 & 0 & 0 & 0 & 0 & 0 & -d_6 & 0 & 0 & 0 & c_0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -d_7 & c_1 & 0 & 0 & c_3 & 0 & 0 & 0 \\
0 & 0 & 0 & c_2 & 0 & 0 & 0 & 0 & -d_8 & 0 & 0 & 0 & c_3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -d_9 & c_1 & 0 & 0 & c_3 & 0 \\
0 & 0 & 0 & c_2 & 0 & 0 & 0 & 0 & 0 & 0 & -d_{10} & 0 & 0 & 0 & c_3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -d_{11} & c_1 & 0 & 0 \\
0 & 0 & 0 & 0 & c_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -d_{12} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -d_{13} & c_1 \\
0 & 0 & 0 & 0 & c_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -d_{14}
\end{bmatrix}$$

Also in this case, it is possible to compute the corresponding stationary distribution Π_2 and the throughput R_2. Unfortunately, here the structure of the transition system (and therefore of the generator matrix) is much more complex: it presents several paths with non-empty intersection. As a consequence, without a suitable instantiation of parameters, Π_2 is unpractical to represent. Even with a very rough instantiation of these parameters, the quantitative results show that the effort made by the attacker to break the WEP algorithm is quite limited. Again, to implement cryptography in wireless networks, designers should adopt algorithms that offer more security guarantees, e.g. AES [9], even though this choice has a strong impact on performance.

Acknowledgments

We are deeply indebted with Francesco Romani for his patient guide in the matrix world.

References

[1] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols - The Spi calculus. *Information and Computation 148*, 1:1–70, January 1999.

[2] W.A. Arbaugh and N. Shankar and Y.C.J. Wan Your 802.11 Wireless Network has No Clothes. To appear in *IEEE Wireless Communication Magazine*, 2002.

[3] C. Bodei, M. Buchholtz, P. Degano, F. Nielson, and H. Riis Nielson. Automatic validation of protocol narration. In *Proc. of CSFW'03*, pages 126–140. IEEE, 2003.

[4] C. Bodei, M. Buchholtz, M. Curti, P. Degano, F. Nielson, and H. Riis Nielson, C. Priami. Performance Evaluation of Security Protocols specified in LySa. *Proc of 2nd Workshop on Quantitative Aspects of Programming Languages*, ENTCS, 2004.

[5] N. Borisov, I. Goldberg and D. Wagner. Communications: The insecurity of 802.11. *Proc. of Conference on Mobile Computing and Networking*, 2001.

[6] M. Buchholtz, F. Nielson, and H. Riis Nielson. A calculus for control flow analysis of security protocols. *International Journal of Information Security*, To appear.

[7] I. Cervesato Fine-Grained MSR Specifications for Quantitative Security Analysis, *WITS'04*, pp. 111-127, 2004.

[8] G. Clark. Formalising the specifications of rewards with PEPA. In *Proc. of PAPM'96*, pp. 136-160. CLUT, Torino, 1996.

[9] J. Daemen and V. Rijndael. *The design of Rijndael*. Springer-Verlag, 2002.

[10] P. Degano and C. Priami. Non Interleaving Semantics for Mobile Processes. *Theoretical Computer Science*, 216:237–270, 1999.

[11] P. Degano and C. Priami. Enhanced Operational Semantics. *ACM Computing Surveys*, 33, 2 (June 2001), 135-176.

[12] D. Dolev and A. Yao. On the security of public key protocols. *IEEE T.I.T.*, IT-29(12):198–208, 1983.

[13] H. Hermanns and U. Herzog and V. Mertsiotakis. Stochastic process algebras – between LOTOS and Markov Chains. *Computer Networks and ISDN systems* 30(9-10):901-924, 1998.

[14] J. Hillston. *A Compositional Approach to Performance Modelling.* Cambridge University Press, 1996.

[15] R, Howard. *Dynamic Probabilistic Systems: Semi-Markov and Decision Systems*, Volume II, Wiley, 1971.

[16] J.T. Kohl and B.C. Clifford. *The Kerberos network authentication service (V5)* The Internet Society, Sept. 1993.RCF 1510.

[17] C. Meadows. A cost-based framework for analysis of denial of service in networks *Journal of Computer Security*, 9(1/2), pp.143 - 164, 2001.

[18] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes (I and II). *Info. & Co.*, 100(1):1–77, 1992.

[19] C. Nottegar, C. Priami and P. Degano. Performance Evaluation of Mobile Processes via Abstract Machines. *IEEE Transactions on Software Engineering*, 27(10), 2001.

[20] A. Perrig and D.Song. A First Step towards the Automatic Generation of Security Protocols. *Proc. of Network and Distributed System Security Symposium*, 2000.

[21] C. Priami. Language-based Performance Prediction of Distributed and Mobile Systems *Information and Computation* 175: 119-145, 2002.